Universität St.Gallen – Hochschule für Wirtschafts-,
Rechts- und Sozialwissenschaften, Internationale Beziehungen und Informatik (HSG)

Predicting the Severity of Car Accidents in Zurich City

University of St.Gallen

3,580: Workshop Fundamentals of Data Science

Autumn Semester 2023

Lyudmila Grigoryeva
Jonathan Chassot
Hannah Busshoff

Elia Locher | 21-941-000
Gabriel Oget | 22-610-893
Marco Molnar | 21-917-315

Link to our notebook and sources:

https://universitaetstgallen-my.sharepoint.com/:f:/r/personal/marco_molnar_student_unisg_ch/Documents/DSF2023/Datasets?csf=1&web=1&e=qtqBik

10. Dezember 2023

Table Of Contents

List of figures

1 Introduction

Using data from the city of Zurich on traffic accidents between 2012 and 2022, we want to create a classification model that can make predictions about the severity of an individual accident. More precisely, we distinguish between two classes of traffic accidents. On the one hand, accidents with personal injury (injuries and fatalities) and accidents with solely property damage. The ratio between the two classes is approximately 3:1. One could imagine our model to be used in self-driving cars, or traffic camera systems that detect an accident has occurred and would be able to immediately contact emergency services in case of a high probability for personal injuries.

2 Data Preprocessing and Feature Engineering

First, we would like to mention that we carry out the data preprocessing and feature engineering before the data analysis, as we can then analyze our newly created features with our label in a next step.

Eight datasets are used to determine the severity of a traffic accident. The main dataset consists of road traffic accidents in the city of Zurich from 2012 to 2022. Using the hour of the accident event and the accident location (Swiss Coordinate System - LV95), additional features are extracted from the remaining seven datasets and added to the main dataset. Where necessary, we will not only consider the data at the time of the accident, but also data prior to the time of the accident. In this way, we enable our model to recognize certain patterns and thus improve the prediction. As a result, we obtain a dataset with many features that might be relevant to the outcome of an accident.

2.1 Date-Time

Firstly, we needed to calculate the exact date of each accident, as only the year, month, hour and day of the week were provided. Inspecting the data made it clear, that it was ordered chronologically, enabling us to calculate the day of the month and create Date-Time feature. The implemented function assumes at least one accident occurrence per week, which aligns with the dataset's characteristics—comprising around 55,000 accidents over approximately 4,000 days. The Date-Time column is derived by iterating through the dataset, considering changes in weekdays, months, and years. The resulting Date-Time values serve as a valuable temporal feature for further feature implementation. At the same time the month, weekday, and hour attributes are kept as features which might offer valuable temporal insights for the model.

2.2 Coordinates

Instead of using the given East and North coordinates, we used a list comprehension that formats them into Point objects, which are very useful for some of the calculations later. Notably, all used datasets involving geospatial information adhere to a specific Swiss coordinate reference system, mitigating concerns related to different systems.

2.3 Traffic Volume

We made extensive efforts to include traffic volume data, measured by almost two hundred counting stations all over Zurich City. Utilizing multithreading for efficiency, the code fetches traffic data from 2012 to 2023. The 'prepare_traffic_data' function orchestrates the integration of traffic data into the main dataset, aligning all observations based on their respective Date-Time. Imputation techniques, including calculating group averages to handle missing values, are employed for robust data integration. Furthermore, the code implements filtering mechanisms to eliminate stations with insufficient data, ensuring the dataset's reliability.

Taking the lagged data for the three closest counting stations, we assigned the traffic volume as well as the respective distance to each accident. This way our model might detect that smaller distances to counting stations increase their relevancy and vice versa. The resulting dataset enables our model to detect traffic-related patterns and their correlation with accident severity.

2.4 Weather

We also integrated the data of the three weather stations in Zurich City for the relevant time period. In a similar manner as before, we create lagged features while correcting for the time shift due to daylights savings time. Originally the data collected between march and October was reported one hour earlier to maintain a constant time dimension. As the accidents in our main data frame and all other time-related data is labeled with daylights saving hours and because it is very intuitive, we decided to continue forward in this way. We also had to impute some data. Considering that the longest continuous period of missing values spanned only over about 20hours, we decided to use a simple forward fill, given the high dependency of weather data on past states and its relative unpredictability.

2.5 Public Transport Stops

This dataset was used to calculate the distance to the closest public transport stops and create Boolean values for the kind of vehicle using it (e.g., bus, train, etc.). This was done creating and using point objects to calculate the relative positions, utilizing spatial indexing. It should be mentioned that the dataset contained some public transport stops without any assigned vehicles, which as we found out represented mainly fairy stops at lake Zurich. We decided to fill their vehicle values with False, so the model might detect special patterns around these stops where all vehicle values are False.

2.6 Traffic Zones

In a next step we assigned each accident to its traffic zone, if it occurred in one. This allows the model to map patterns onto certain areas of Zurich city, as well as to certain zone types. Here we had to check the validity of each geometry in the dataset and correct all invalid geometries. Then we iterate over each zone polygon and check whether each accident in the main dataset falls within the spatial and temporal boundaries of the zone, as some zones where implemented during the time period of our accident dataset. Lastly, the code fills any missing values with a 'no_zone' string.

2.7 Road Network and Speed limit

Given the geometry of the entire Zürich City road network, we were able to assign each accident to its closest street and its speed limit. Furthermore, our code detects all possible intersections, by counting the occurrence of each point in the road network. All streets that meet at an intersection contain the same point. This enabled us to calculate the distance to the next intersection in both directions of the accident, following the actual road network.

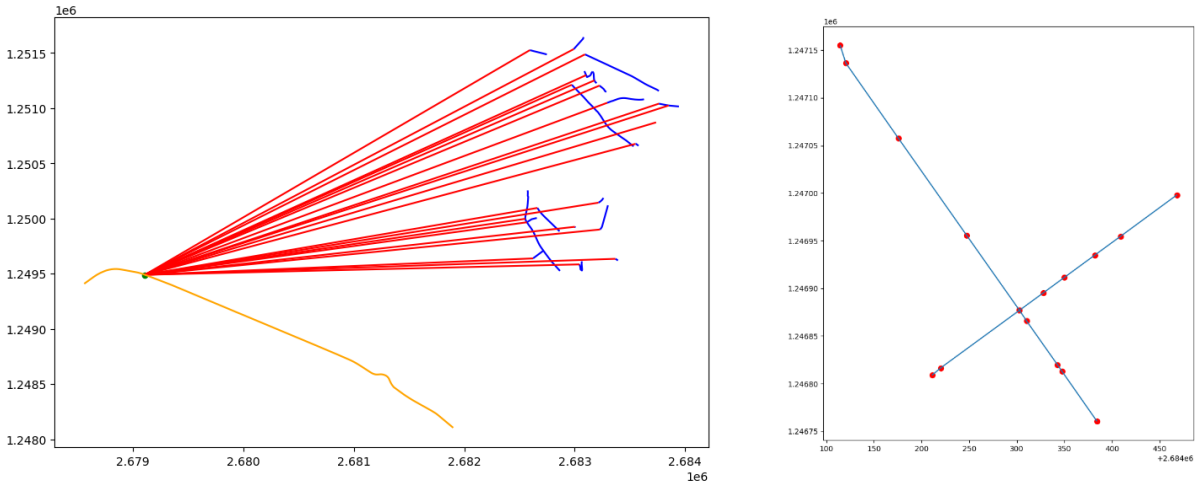

Figure 1: Closest distance to street network and common points at intersections

(self-created figure)

2.8 Pedestrian and Bicycle Network

Similar to the road network, we were also able to obtain the geometries of the entire pedestrian and bicycle pathway in Zurich City. The code identifies pedestrian crossings in Zurich by comparing footpath and street geometries. It utilizes the Shapely library to find crossing points where footpaths and streets intersect. These crossings are stored in a dictionary, with keys representing intersection points and values denoting the corresponding street index. Subsequently, the code calculates distances from accidents to the nearest bicycle paths, considering the kind of bicycle path. The results are incorporated into the dataset, enriching it with details about accident proximity to pedestrian crossings and bike infrastructure.

2.9 Development of the Feature Engineering Process

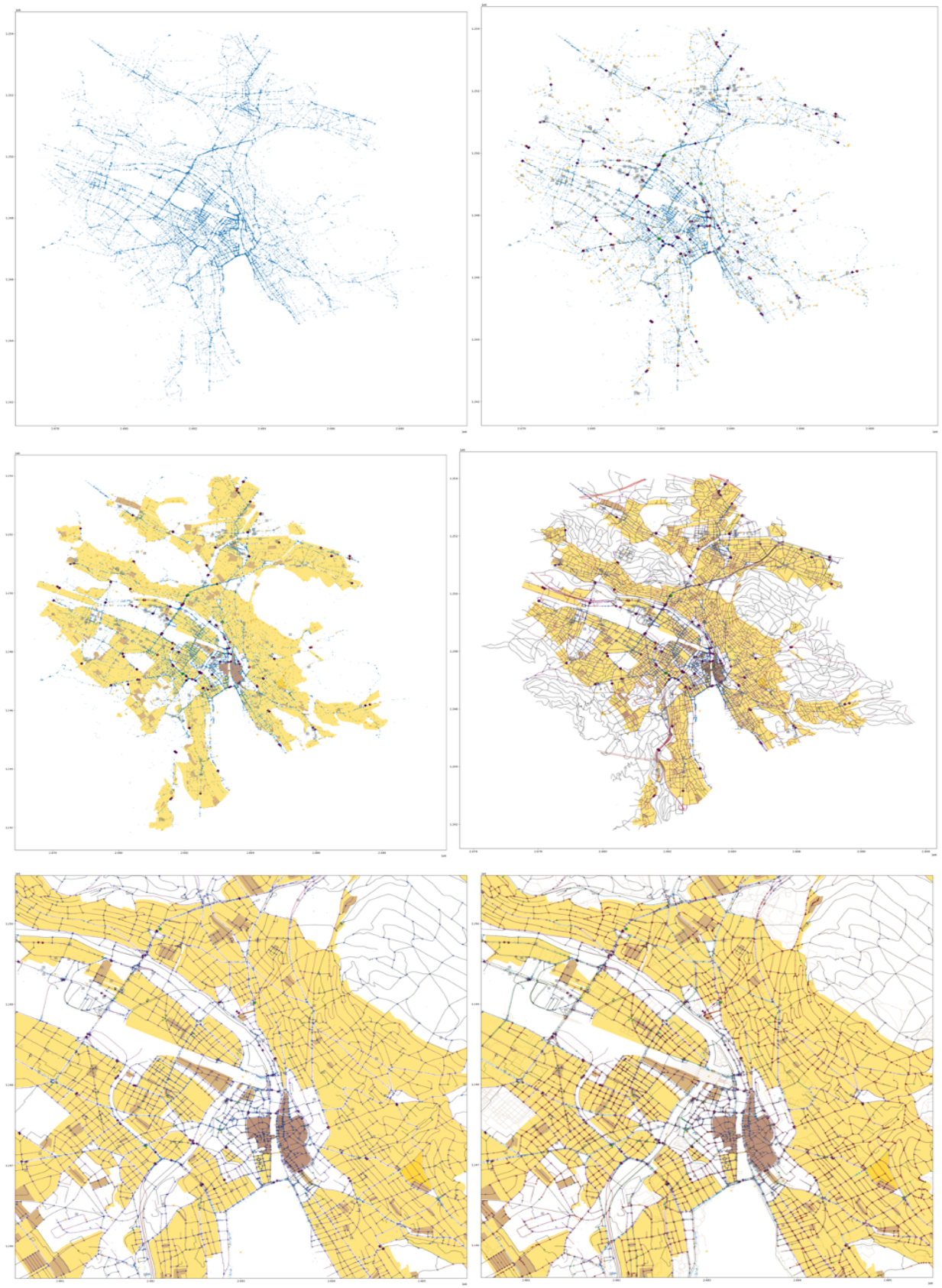

Figure 2: Development of the feature engineering process

(self-created figure)

2.10 Holidays in Zurich City

Finaly we implemented a simple Boolean feature that indicated, wether the he accident occurred on a public federal, cantonal or regional holiday. The provided code defines and populates lists of fixed-date holidays in Zurich from 2012 to 2022, including events like ‘Sechseläuten’ and ‘Knabenschiessen’. It uses the `date` class to represent holiday dates and calculates variable-date holidays based on the Easter date for each year. This information can be valuable for analyzing accident patterns and understanding potential correlations with holiday events.

3 Data Analysis

3.1 Bar Charts

First we took the features that we already had in our original dataframe. We compared the characteristics of a feature such as the different accident types or the datetime with the relative distribution of our label. We use different bar charts because, although they are simple, they provide a quick and good overview.

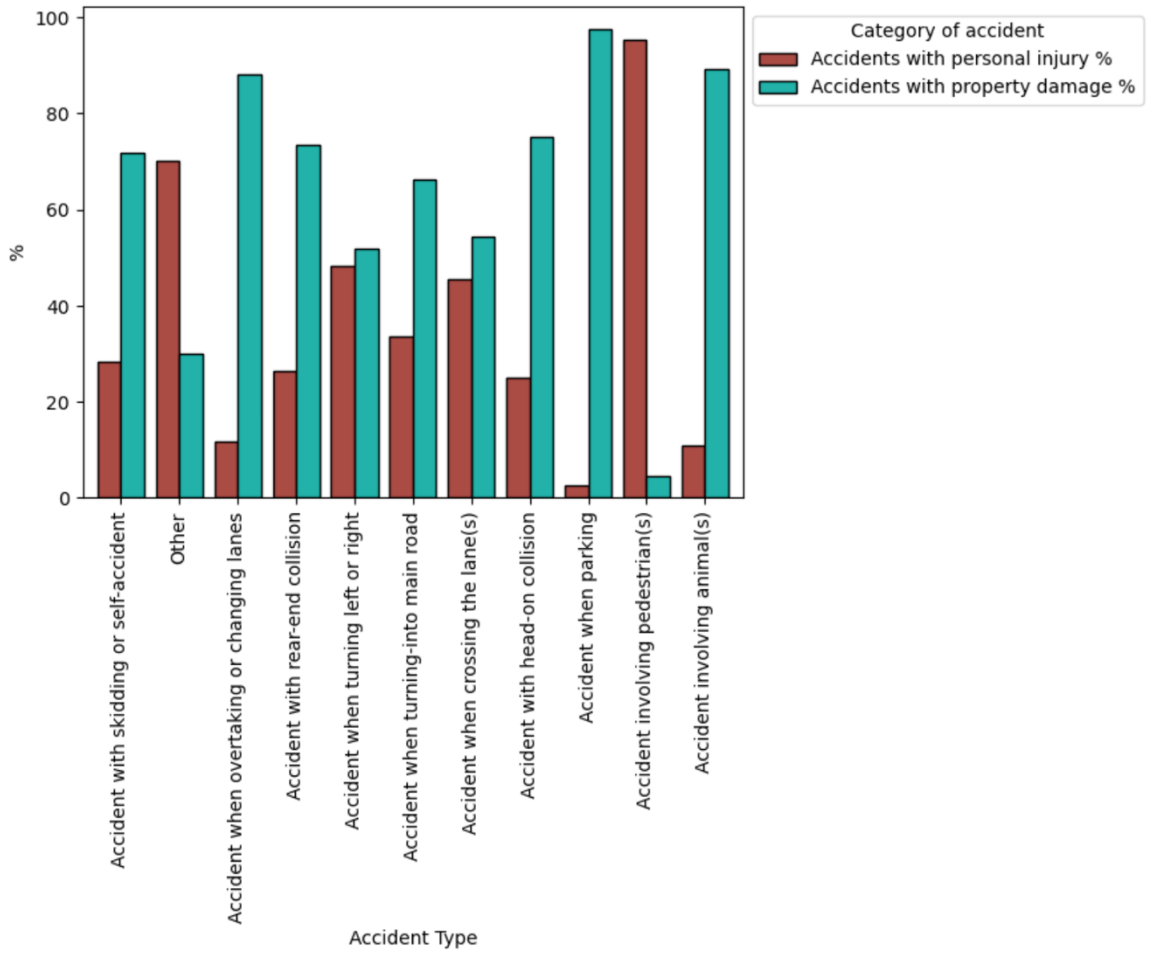

Figure 3: Bar chart for the different accident types

(self-created figure)

3.2 Correlation Matrices

We also used correlation matrices to analyze the dependency between our weather features and our label on the one hand and the dependency between our traffic volume and our label. In this case, it was important to keep in mind that a correlation implies a dependence, but a dependence does not imply a correlation.

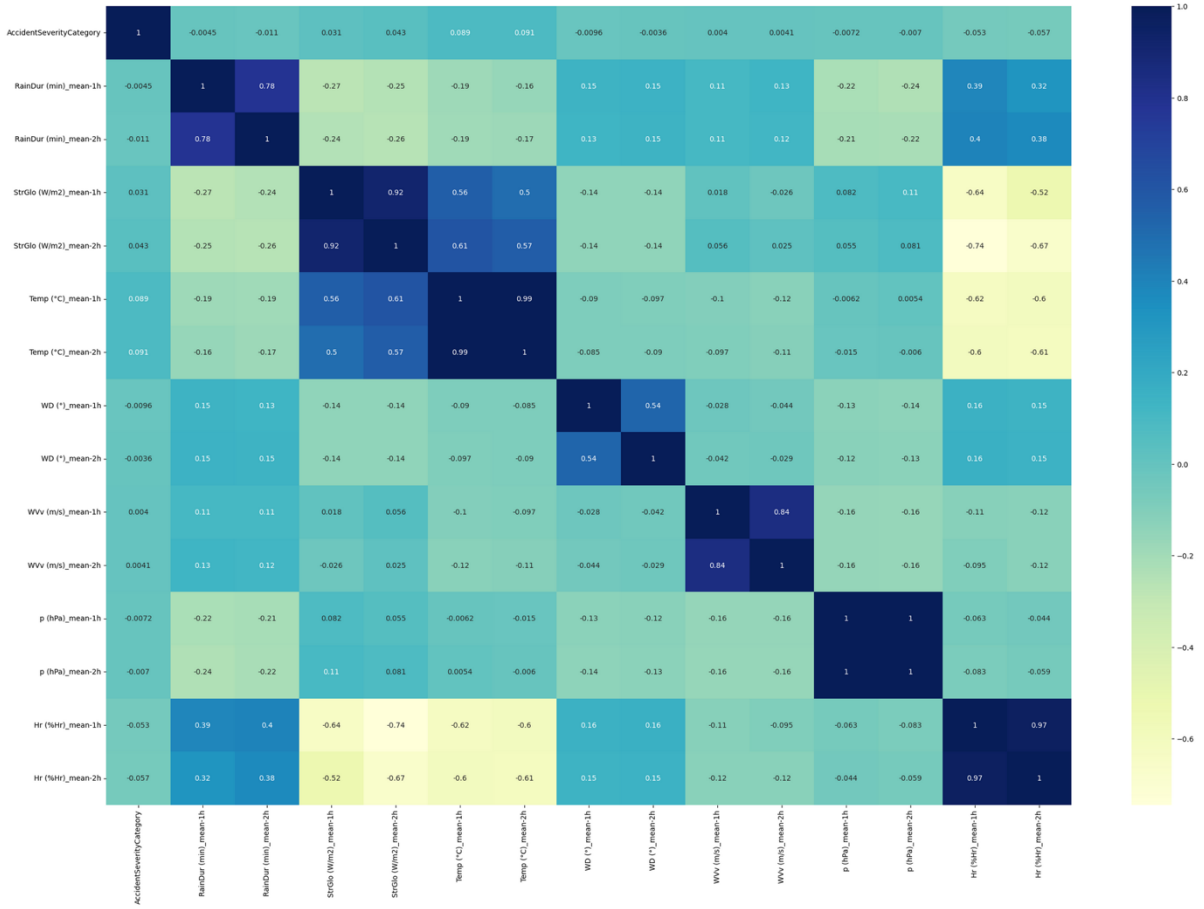

Figure 4: Correlation matrix for the weather data

(self-created figure)

4 Model Training

First, our dataset underwent a crucial split into a training set and a test set. We then decided to divide our dataset again so that we have a train, validate and test set. We chose this because we should consider our test set as 'future data' which we have 'never seen before'. The fact that we have a sufficiently large data set with over 50 thousand observations also spoke in favor of this two-part split.

This process also allows us to train our models on the training data, tune hyperparameters, evaluate their performance against an unknown validation set, and only use the test set at the very end to draw final conclusions.

By adopting this approach, we emulate real-world scenarios and ensure that our model selection and hyperparameter tuning decisions are based on their generalization to novel, unobserved data. This rigorous methodology avoids coming into contact with the actual test set until the final stage, ensuring that its performance serves as an unbiased assessment.

4.1 Finding Hyperparameters

To find our best hyperparameters, we installed the function *RandomizedSearchCV* and *GridSearchCV* from the *sklearn.model_selection* library. We discussed the advantages and disadvantages of both methods in detail in our notebook.

4.2 Our Model Selection

We wanted to train a wide range of models, from simple ones like naive Bayes to very complex but also interesting ones like a neural network. What was important here, however, was that we first looked at which classification metrics we wanted to optimize. As we have documented in detail in our notebook, recall is very crucial for our use case. We would like to reduce the number of false negative results, as it can have fatal consequences if personal injury (a positive) is classified as property damage (a negative).

4.2.1 Random Forest Classifier

We have different random forest classifiers, from a very simple one to a model that gives us the best hyperparameters through hyperparameter tuning and cross-validation. It was interesting to see that the simple random forest classifier already performed well in the validation set with regard to the Area Under the Curve" of the “Receiver Operating Characteristic” curve.

4.2.2 Logistic Regression

In addition to our different random forest models, we also used logistic regression. In a first step we trained a simple logistic model. We used for regularization *Lasso* and *Ridge*. We also found the optimal hyperparameters with *RandomizedSearchCV*.

4.2.3 Neural Network

The idea was initially to become a little familiar with neural networks. We used the *Keras* library to build our neural network. As a first step, we tried to find out how the network reacts with different numbers of neurons per hidden layer. We also tried playing around with the number of hidden layers. The first thing we noticed was that the networks with more hidden layers and more neurons per layer did not perform any better than the computationally efficient smaller networks. During this trial and error, we kept all the hyperparameters the same. This approach is very inefficient and ineffective. For this reason, we trained our model using the *Kerastuner* library from *Keras*. As we already mentioned and go deepter into it in our notebook, we used *RandomizedSearchCV* because we read that this method is the most cost-effective way to check different hyperparameters. We applied this method several times with different intervals and steps of the parameters. We used the three best models to predict the validation dataset. These results of the neural networks serve as a basis for the comparison with the other models and ultimately for the decision for the best model.

5 Conclusion

Certain models such as a basic decision tree or the naive bayes classifier performed very poorly on the validation set. In the remaining models, namely random forest, logistic regression and our neural network, the difference is minimal. Here we plot an evaluation for the random forest classifier.

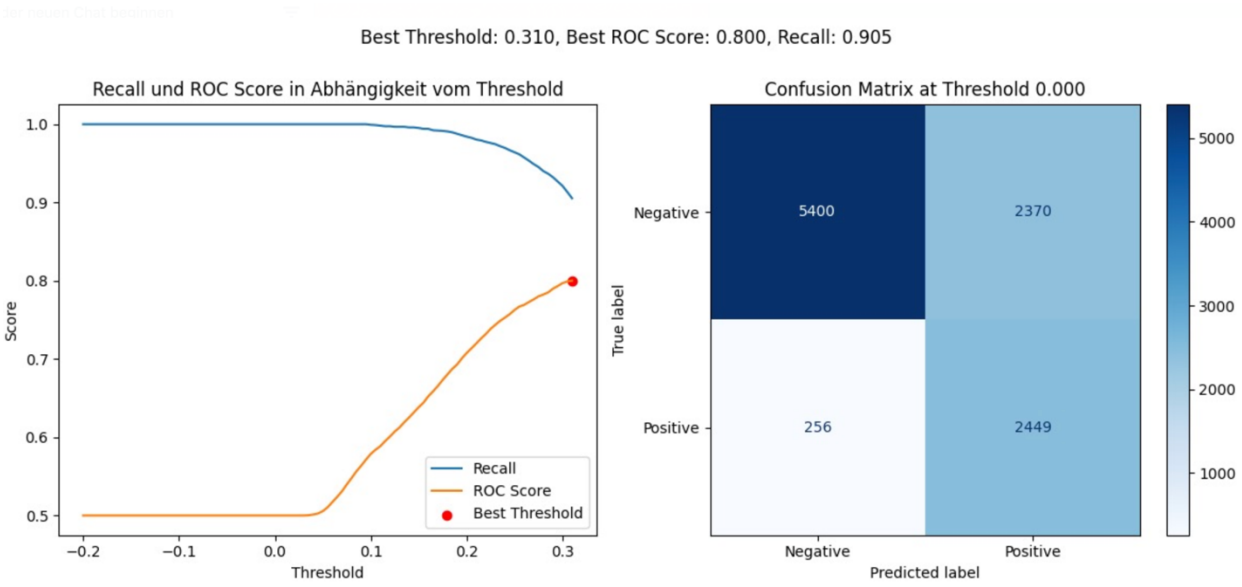

Figure 5: Results of (one of) our best model

(self-created figure)

Appendix

Description	Variables	Access
Main Dataset: Shows traffic accidents in the city of Zurich from 2012 to 2022	**Label** *Accident severity:* • Accident with property damage • Accidents with personal injury **Features** • *Accident type* (e.g. Accident with skidding or self-accident) • *Parties involved in the accident* (e.g. Bicycle, Pedestrian or Motorcycle) • *Road type* (e.g. Minor Road, Principal Road or Other) *Accident date* and *Accident location* are used to create other features.	Accident Data (CSV-Datei)
Weather Data: Annual hourly measured weather data from 2012 to 2023 for the city of Zurich at three measuring stations. We select the weather data from the measuring station closest to the scene of the accident (correction: select the mean of each variable over the three weather stations)	**Features** • Air pressure (p), • Duration of precipitation (RainDur), • Global radiation (StrGlo), • Temperature (T), • Relative humidity (Hr), • Wind direction • Vector and scalar wind speed	Weather Data
Traffic volume Data:	**Feature**	Traffic Volume (motorized individual traffic)

Hourly measured number of vehicles (motorized individual traffic) from 2012 to 2023 at various measuring stations in the city of Zurich. We select the traffic volume of the nearest measuring station for each accident.	• Traffic volume	
Frequency of bicycle and pedestrian traffic: This dataset provides information on the frequency of bicycle and pedestrian traffic. Again, we will use the values from the nearest counting station to the accident	**Features** • Number of pedestrians at the nearest measuring station • Number of cyclists at the nearest measuring station	Frequency of bicycle and pedestrian traffic
Pedestrian and cycle path network: Includes the entire walking and cycling network of the city of Zurich. We want to find out the distance between the accident and the nearest pedestrian crossing, as well as the type of cycle path for accidents involving bicycles	**Features** • Distance of the accident to the nearest pedestrian crossing • Type of cycle path	Pedestrian and cycle path network (gpkg-Datei)
Location bus and streetcar stops: Location of all bus and streetcar stops in the city of Zurich. We want to find out the distance of the accident to the nearest bus stop.	**Feature** • Distance between the accident and the nearest bus or streetcar stop	Bus and streetcar stops (CSV-Datei)
Traffic areas: Shows the existing traffic zones in the city of Zurich (e.g. meeting zone, pedestrian zone, T30 zone). We also take into account when the traffic zone was implemented. We want to create dummy variables to find out whether an accident occurred in a T-30 zone, a pedestrian zone or a meeting zone.	**Feature** • Dummy Variable of the traffic zone in which the accident occurred (meeting zone, pedestrian zone, T30 zone)	Traffic areas (gpkg-Datei)

Signalized speed regimes: **This dataset contains the entire road network of the city of Zurich. Here, too, we take into account the date of implementation of newly built roads.** Our aim is to extend the above dummy variables by T20, T50, T60, T80, T100, T120 zones. In addition, we will find out the distance of the accident to the next junction.	**Features** • Dummy Variable of traffic zone in which the accident occurred (T20, T50, T60, T80, T100, T120) • Distance of the accident to the next junction	Signalized speed regimes (CSV-Datei)